RODANDO OS PRIMEIROS QUBITS

```
from azure.quantum import Workspace
workspace = Workspace (
  resource_id = "",
  location = "eastus"
)
```

Importando o Qiskit:

```
from qiskit import *
```

Criando os dois registradores quânticos, isto é, os quatro bits quânticos.

```
qr = QuantumRegister(4)
```

Criar os dois registradores clássicos, isto é, os quatro bits binários.

```
cr = ClassicalRegister(4)
```

Agora que temos dois registradores quânticos e dois clássicos, podemos criar um circuito.

```
circuit = QuantumCircuit(qr, cr)
```

Criamos um circuito quântico, em qualquer ponto que nós modificarmos o circuito, podemos visualizar ele com:

```
%matplotlib inline
circuit.draw()
```

Saída:

q0_0:

q0_1:

q0_2:

q0_3:

c0: 4/

Foi necessário instalar o pylatexenc para darmos andamento no gate

pip install pylatexenc

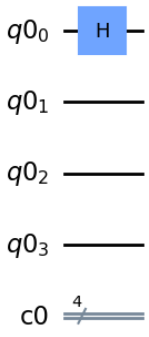
Defaulting to user installation because normal site-packages is not writeable Requirement already satisfied: pylatexenc in ./.local/lib/python3.9/site-packages (2.10) **[**notice**]** A new release of pip is available: 23.0.1 -> 23.1 **[**notice**]** To update, run: pip install --upgrade pip Note: you may need to restart the kernel to use updated packages.

Então vamos criar um gate.
Para criar esse emaranhamento quântico, o primeiro passo é aplicar o que é chamado de hadamard gate no seu primeiro qubits. O que vamos fazer então é criar um circuito hadamard e aplicar ao primeiro qubits.

E plotar a saída usando o matplotlib.

```
circuit.h(0)
circuit.draw(output='mpl')
```

Saida:




```
circuit.cx(0,1)
```

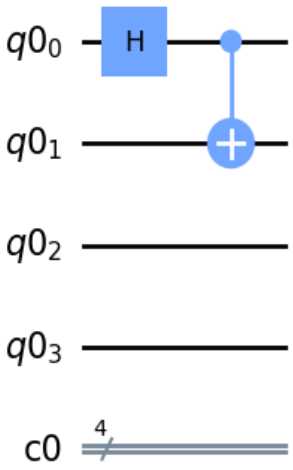
Saida:

<qiskit.circuit.instructionset.InstructionSet at 0x7fa468706b50>

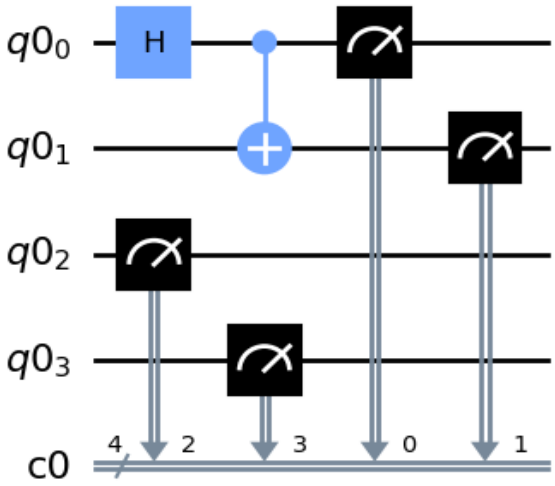
Vamos desenhar o circuito com:

```
circuit.draw(output='mpl')
```

Saida:




```
circuit.measure(qr, cr)
circuit.draw(output='mpl')
```



O circuito agora tem um hadamard gate e um controle.

A ideia é que com essas duas operações simples, nós seremos capazes de gerar emaranhamentos em bits quânticos q0_0, q0_1.

Então, agora que criamos nosso circuito quântico, usando o hadamard gate e o controle x, o que vamos fazer agora é medir o bit quântico, pegar sua medição e armazenar em um bit clássico.

Nosso circuito quântico está da forma mostrada acima.

Temos uma operação quântica, o **hadamard gate** e o **controlled x gate** e temos a medição.

O próximo passo é executar o circuito, e iremos fazer duas coisas:

1. Executar o circuito em um computador clássico e ver o que acontece quando simulamos um computador quântico.
2. Executar em um dispositivo quântico real da IBM e então observar os resultados retornados.

Para simular o circuito quântico, o que vamos fazer é importar o componente Aer do Qiskit.

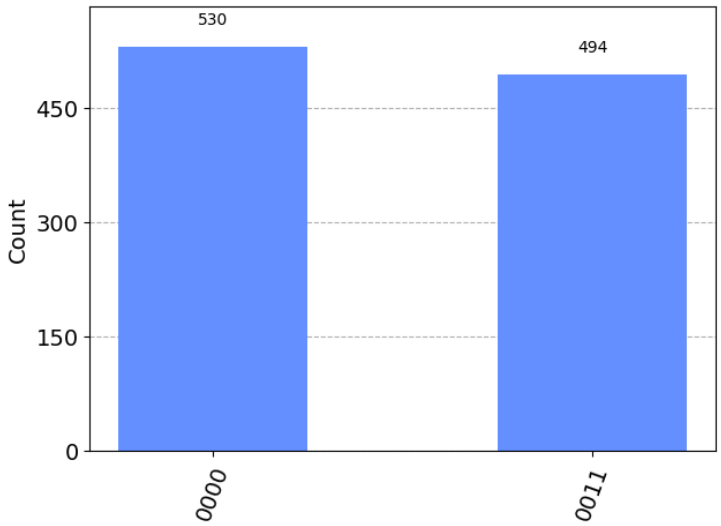
A ideia é usar o Aer no nosso computador local para simular o circuito quântico.

```
simulator = Aer.get_backend('qasm_simulator')

result = execute(circuit, backend=simulator).result()

from qiskit.visualization import plot_histogram

plot_histogram(result.get_counts(circuit))
```



Tivemos aproximadamente 50% ou 0.5 de probabilidade de zero zero e quase 50% também de um um.

Esse pequeno erro é porque estamos executando um limitado número de tentativas em nossa simulação ao invés de infinitas.

EXEMPLO PARA IMPLEMENTAÇÃO DO BANCO DE DADOS EM PYTHON

Utilizamos um banco de testes

$$import\ mysql.connector$$

Configurações de conexão com o banco de dados

$$config\ =\ \{$$

$$'user': 'seu\_usuario',$$

$$'password': 'sua\_senha',$$

$$'host': 'localhost',$$

$$'database': 'seu\_banco'$$

$$\}$$

Cria a conexão com o banco de dados

$$conexao\ =\ mysql.connector.connect(*config)$$

Cria um cursor para executar consultas SQL

$$cursor\ =\ conexao.cursor()$$

Executa uma consulta SQL

$$cursor.execute('SELECT\ FROM\ minha\_tabela')$$

Exibe o resultado da consulta

$$for\ linha\ in\ cursor:$$

$$print(linha)$$

Fecha o cursor e a conexão com o banco de dados

$$cursor.close()$$

$$conexao.close()$$